基于朴素贝叶斯与 SVM 的垃圾邮件分类模型的实现与评估

数据挖掘算法与应用实验报告

2019 秋季学期

1. 问题描述

垃圾邮件检测是机器学习在现今互联网领域的主要应用之一。几乎所有大型电子邮箱服务提供商都内置了垃圾邮件检测系统，能够自动将此类邮件分类为“垃圾邮件”。

本实验中的主要任务是，通过朴素贝叶斯算法和 SVM 算法训练出垃圾邮件分类模型，再使用 Bootstrap 方法、K-Fold 交叉验证法、以及绘制 ROC 曲线对两个模型进行评估，进而对两个模型进行比较。

2. 问题分析及解决方案

2.1. 问题的形式化描述

垃圾邮件分类问题可以看作一个二分类问题：

训练阶段：

输入：训练集$D_{train} = (X_{train}, y_{train})$，其中，$X_{train} = (x_1, x_2, \dots, x_n)$为训练集中每封邮件的词向量，$y_{train} \in \{0, 1\}$为训练集给出的邮件类别

输出：垃圾邮件分类模型 M

测试阶段：

输入：垃圾邮件分类模型 M；测试集$D_{test} = (X_{test}, y_{test})$，其中，$X_{test} = (x_1, x_2, \dots, x_n)$为测试集中每封邮件的词向量，$y_{test} \in \{0, 1\}$为测试集给出的邮件类别

输出：测试集中每封邮件的预测类别$y_{predict}$

本实验采用朴素贝叶斯算法和 SVM 算法，由于两种算法选取的特征不同，所以每封邮件的词向量的表示方法也不同，具体内容在 2.2 节中进行描述。

2.2. 问题的解决方案

2.2.1. 朴素贝叶斯分类模型

词向量的表示方法：

统计数据集中出现的所有单词，生成词库。再统计词库中的每个单词在一封邮件中出现的次数作为该邮件的词向量。

假设词库$\mathrm{V}=\{v_1,v_2,\ldots,v_n\}$，其中$v_i(i=1,2,\ldots,n)$表示在数据集中出现过的单词，则每封邮件的词向量可以表示为$X^{(j)}=(x_1,x_2,\ldots,x_n)(\mathrm{j}=1,2,\ldots,\mathrm{m})$，其中$x_i(i=1,2,\ldots,n)$表示单词$v_i$在邮件$j$中出现的次数。

先验概率及后验概率的计算：

根据朴素贝叶斯算法的原理，可知：

$$y_{predict}=\underset{c_i}{\operatorname{argmax}}P(c_i)\prod_{j=1}^{n}P(x_j|c_i),c_i\in\{0,1\}$$

在本实验中：

$$P(c_i)=\log(\frac{\#(c_i)}{m})$$

$$\mathrm{P}(x_j|c_i)=\log(\frac{sum\left(x_j^{c_i}\right)+1}{sum(x^{c_i})+n})$$

其中，$\#(c_i)$表示垃圾邮件的数目，$sum\left(x_j^{c_i}\right)$表示在垃圾邮件中单词$x_j$出现的次数，$sum(x^{c_i})$表示在垃圾邮件中所有单词出现的总次数。

为避免后验概率的结果为 0，进而导致最终得到的联合概率为 0，使用拉普拉斯平滑；为避免由于概率值太小而导致向下溢出，取概率的对数值作为结果，取对数不改变原本数值的大小关系，此时联合概率变为：

$$y_{predict}=\underset{c_i}{\operatorname{argmax}}P(c_i)+\sum_{j=1}^{n}P(x_j|c_i)$$

2.2.2. SVM 分类模型

词向量的表示方法：

统计训练集的垃圾邮件中出现的单词及其出现的次数，将出现次数大于等于 k 次的单词放入词库。再统计词库中的每个单词在一封邮件中是否出现作为该邮件的词向量。

假设词库$\mathrm{V}=\{v_1,v_2,\ldots,v_n\}$，其中$v_i(i=1,2,\ldots,n)$表示在训练集的垃圾邮件中出现次数大于等于 k 次的单词，则每封邮件的词向量可以表示为$X^{(j)}=(x_1,x_2,\ldots,x_n)(\mathrm{j}=1,2,\ldots,\mathrm{m})$，其中$x_i(i=1,2,\ldots,n)$表示单词$v_i$在邮件$j$中是否出现，若出现则置为 1，否则置为 0。

2.3. 结果的评估

2.3.1. Bootstrap 方法

从数据集中有放回地随机取出 n 个样本，去重后得到该数据集的训练集，剩余部分则作为测试集。

在每次随机采样中，一个样本没有被选取的概率为$1-\frac{1}{n}$，则 n 次都没有被选取的概率为$(1-\frac{1}{n})^n$，当 n 趋近于无穷大时，这个概率趋近于$\frac{1}{e}\approx 0.368$。

由 Bootstrap 方法，将数据集划分为了约占 63.2%的训练集和约占 36.8%的测试集。

2.3.2. K-Fold 交叉验证法

将数据集平均划分为 k 等份，轮流将 k-1 份作为训练集，剩余的 1 份作为测试集，进行 k 次训练和测试，每次都计算模型的 accuracy、precision、recall 以及 F1 值，最终计算四个度量的平均值作为模型的真实评估结果。

2.3.3. ROC 曲线

通过计算得到模型将每个样本预测为正样本的概率，改变 threshold，得到一组真阳性率（TPR）和假阳性率（FPR）。以假阳性率为横轴，真阳性率为纵轴绘制曲线，即为模型的 ROC 曲线，ROC 曲线下方的面积即为模型的 AUC 值。

3. 实验过程及结果分析

3.1. 实验环境

Python 3.7.3

PyCharm 2019.1.3 (Community Edition)

Windows 10

3.2. 实验数据

数据来源：http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

数据描述：

SMS Spam Collection 是一组公开的带有标签的 SMS 邮件数据，从网上的免

费研究资源中收集得到，共包含 5574 条 SMS 邮件数据，其中 747 条为垃圾邮件，4827 条为非垃圾邮件。

数据属性：

SMS Spam Collection 仅有一个文件，文件中每行为一条邮件数据，每条数据包含两列信息：第一列为邮件标签，spam 表示垃圾邮件，ham 表示非垃圾邮件；第二列为邮件内容。此外，邮件没有按照时间进行排序。

数据示例：

```
ham Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
ham Ok lar... Joking wif u oni...
spam    Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std
ham U dun say so early hor... U c already then say...
ham Nah I don't think he goes to usf, he lives around here though
spam    FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! Xx
ham Even my brother is not like to speak with me. They treat me like aids patent.
ham As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers.
spam    WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 090617014
spam    Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call T
ham I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
```

3.3. 实验过程

3.3.1. 数据预处理

用于朴素贝叶斯的词向量生成：

按行读入数据，分离标签信息和邮件内容。

处理标签信息：spam 为 1，ham 为 0。

```
def load_sms_data(file_name):
    """..."""
    class_category = []    # 类别标签
    sms_words_list = []    # 文本单词，包含所有行，每行为一个单词列表
    f = open(file_name, 'r', encoding='UTF-8')
    for line in f.readlines():
        line_data = line.strip().split("\t")    # strip删除行两侧的空白字符，split按照空白字符对文本进行分割
        if line_data[0] == "ham":
            class_category.append(0)
        elif line_data[0] == "spam":
            class_category.append(1)
        words = text_parser(line_data[1])
        sms_words_list.append(words)
    f.close()
    return sms_words_list, class_category
```

图 3-1 处理标签信息的相关代码

处理邮件内容：删除行两侧的空白字符，去掉非字母和数字，并将所有字母转换为小写，再将剩余的部分按照空白字符分割，得到一封邮件的单词列表。

```
13  def text_parser(text):
14      """..."""
19      reg_ex = re.compile(r'[^a-zA-Z]|\d')     # 去掉非字母或数字，只留下单词
20      words = reg_ex.split(text)
21      words = [word.lower() for word in words if len(word) > 0]     # 转换为小写
22      return words
```

图 3-2 处理邮件内容的相关代码

生成词库：取所有邮件的单词列表的并集得到词库，即词库中保存的单词是整个数据集中所有可能出现的单词，记词库中单词的数目为 n。

```
46  def create_vocabulary_list(sms_words_list):
47      """..."""
52      vocabulary_list = set([])
53      for sms_words in sms_words_list:
54          vocabulary_list = vocabulary_list | set(sms_words)
55      vocabulary_list = list(vocabulary_list)
56      return vocabulary_list
```

图 3-3 Bayes 模型生成词库的相关代码

生成词向量：为每封邮件生成一个 n 维向量作为该邮件的特征向量，n 维向量中的第 i 个分量表示词库中第 i 个单词在这封邮件中出现的次数。

```
71  def create_words_vector(vocabulary_list, sms_words):
72      """..."""
78      sms_words_vector = [0] * len(vocabulary_list)
79      for sms_word in sms_words:
80          if sms_word in vocabulary_list:
81              sms_words_vector[vocabulary_list.index(sms_word)] += 1
82      return sms_words_vector
```

图 3-4 Bayes 模型生成词向量的相关代码

用于 SVM 的词向量的生成：

处理标签信息和邮件信息的方法同朴素贝叶斯算法。

生成词库：统计垃圾邮件中出现的单词及其出现的次数，将出现次数大于等于 min_times 次的单词放入词库中，记词库中单词的数量为 n。

```
61  def create_vocabulary_list(spam_words_list, times):
62      """..."""
67      spam_words_list = pd.DataFrame(spam_words_list)
68      # 计算每个单词在垃圾邮件中出现的总次数
69      num_spam_words = spam_words_list.apply(pd.value_counts).apply(lambda x: x.sum(), axis=1)
70      vocabulary_list = []
71      for index, value in num_spam_words.iteritems():
72          if value >= times:    # 将出现times次以上的单词记录在词库中
73              vocabulary_list.append(index)
74      return vocabulary_list
```

图 3-5 SVM 模型生成词库的相关代码

生成词向量：为每封邮件生成一个 n 维向量作为该邮件的特征向量，n 维向量中的第 i 个分量表示词库中第 i 个单词在这封邮件中是否出现，若出现则置为 1，否则置为 0。

```
77  def create_words_vector(vocabulary_list, sms_words):
78      """..."""
84      sms_words_vector = [0] * len(vocabulary_list)
85      for sms_word in sms_words:
86          if sms_word in vocabulary_list:
87              sms_words_vector[vocabulary_list.index(sms_word)] = 1
88      return sms_words_vector
```

图 3-6 SVM 模型生成词向量的相关代码

3.3.2. 模型的训练与评估

Simple Evaluate:

在同一训练集上（data_train.txt）分别训练朴素贝叶斯分类模型和 SVM 分类模型，再在同一测试集上（data_test.txt）对两个模型进行评估，评估方法包括计算 accuracy、precision、recall、F1 值和绘制 ROC 曲线。

```
55  accuracy = module.score(test_words_matrix, test_class_category)
56  precision = precision_score(test_class_category, class_result, average='macro')
57  recall = recall_score(test_class_category, class_result, average='macro')
58  f1 = f1_score(test_class_category, class_result, average='macro')
```

```
63  p_list = module.predict_log_proba(test_words_matrix)[:, 1]
64  fpr, tpr, threshold = roc_curve(test_class_category, p_list)      # 计算真阳性率和假阳性率
65  roc_auc = auc(fpr, tpr)
```

图 3-7 计算 accuracy、precision、recall、F1 值、TPR、FPR 的相关代码

对于 SVM 模型，还要选择合适的 min_times 值，因此在 Simple Evaluate 阶段还要对不同的 min_times 值对模型表现的影响进行评估，从而选出最优的 min_times 值。

Bootstrap Evaluate:

载入数据集（data.txt），记样本总数为 m。

对数据集进行划分，生成 m 个[0, m)的随机数，去重后，得到训练集中样本的索引，剩余的样本放入测试集中。

```
33  index_list = [random.randint(0, num_lines) for _ in range(num_lines)]    # 随机生成num_lines个数
34  index_list = set(index_list)          # 去重
35  train_words_matrix = []               # 用于训练的单词矩阵
36  test_words_matrix = []                # 用于测试的单词矩阵
37  train_class_category = []             # 用于训练的类别标签
38  test_class_category = []              # 用于测试的类别标签
39  for i in range(num_lines):
40      if i in index_list:
41          train_words_matrix.append(words_matrix[i])
42          train_class_category.append(class_category[i])
43      else:
44          test_words_matrix.append(words_matrix[i])
45          test_class_category.append(class_category[i])
```

图 3-8 使用 Bootstrap 方法对数据集划分的相关代码

K-Fold Evaluate:

本实验中令 k=10，因此轮流将其中 9 份作为训练集，剩余的 1 份作为测试

集，进行 10 次训练和测试，最终将每轮得到的 accuracy、precision、recall、F1 值的平均值作为模型的真实评估结果。

```
35      test_words_matrix = words_matrix[i * fold_len: (i + 1) * fold_len]     # 用于测试的单词矩阵
36      train_words_matrix = words_matrix[: i * fold_len]                      # 用于训练的单词矩阵
37      train_words_matrix.extend(words_matrix[(i + 1) * fold_len:])
38
39      test_class_category = class_category[i * fold_len: (i + 1) * fold_len]     # 用于测试的类别标签
40      train_class_category = class_category[: i * fold_len]                      # 用于训练的类别标签
41      train_class_category.extend(class_category[(i + 1) * fold_len:])
```

图 3-9 使用 K-Fold 方法对数据集划分的相关代码

3.4. 实验结果

3.4.1. Simple Evaluate

SVM 模型中 min_times 的选取：

在同一训练集和测试集上分别将 min_times 设为不同值，进行模型的训练与评估，得到的结果如图 3-10 所示。由图可知，随着 min_times 的增大，accuracy、precision、recall、F1 值先随之急剧增大，后逐渐减小；通过 ROC 曲线得到的 AUC 值，随 min_times 值的增大而逐渐减小。

综合两个结果，本实验选取 min_times=20。

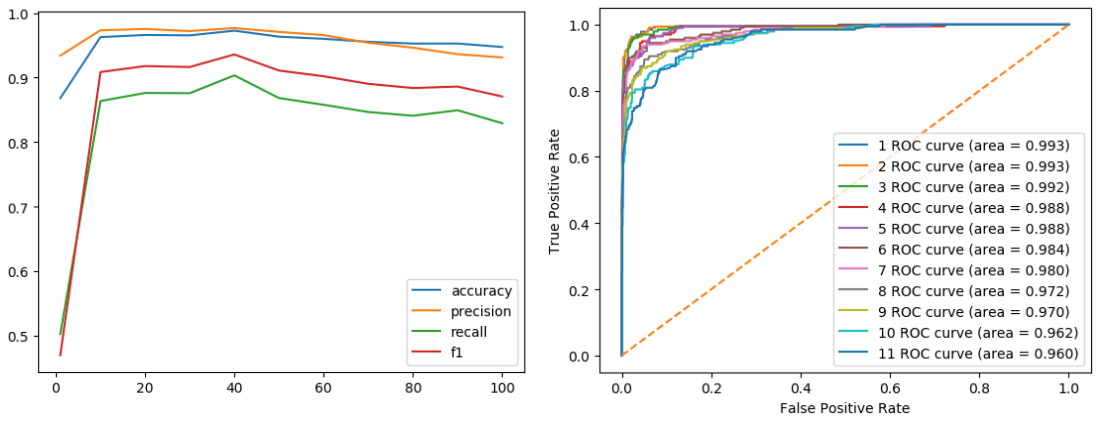

图 3-10 accuracy、precision、recall、F1 值随 min_times 值的变化（左），
ROC 曲线随 min_times 值的变化（右）

朴素贝叶斯模型的表现：

自实现的朴素贝叶斯的表现和 python 库的朴素贝叶斯的表现如图 3-11 所示。由图可知，python 库的朴素贝叶斯模型要比自实现的模型表现好得多。

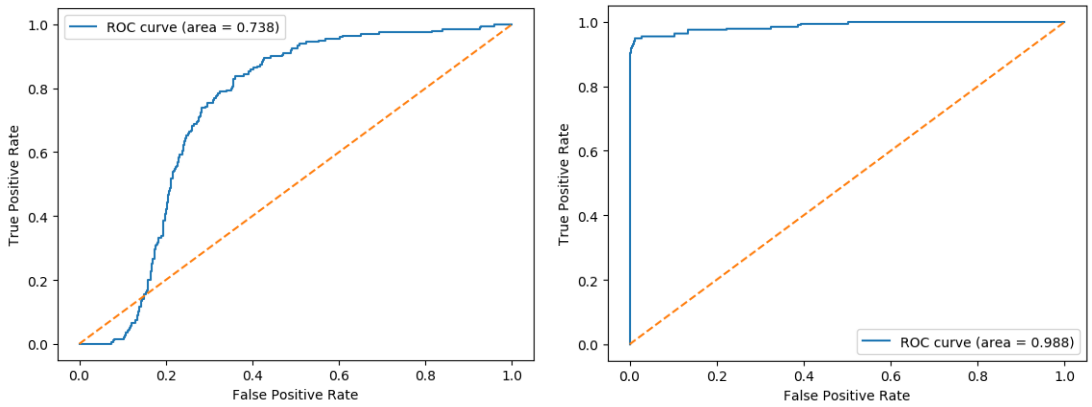

图 3-11 自实现朴素贝叶斯模型的 ROC 曲线（左），

python 库的朴素贝叶斯的 ROC 曲线（右）

朴素贝叶斯模型与 SVM 模型的对比：

由于 SVM 是由 python 库中的 SVM 模型进行训练的，所以同样使用 python 库中的贝叶斯模型进行比较。对于 SVM 模型，选取最优的模型，即 min_times=20 的模型。

由图 3-12 和图 3-13 可知，在同一训练集和测试集下，朴素贝叶斯模型的 accuracy、recall、f1 值都优于 SVM 模型，而 SVM 模型的 precision 略优于朴素贝叶斯模型，总体来说，朴素贝叶斯模型更优。

```
正在执行第 1 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 1 ......
词库中包含 1819 个单词
precision( 1 ) = 0.9339559706470981 , recall( 1 ) = 0.5025125628140703 , accuracy( 1 ) = 0.868 , f1( 1 ) = 0.46964285714285714
正在执行第 2 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 10 ......
词库中包含 253 个单词
precision( 2 ) = 0.973241624809569 , recall( 2 ) = 0.8635529685321304 , accuracy( 2 ) = 0.9626666666666667 , f1( 2 ) = 0.9085250314717451
正在执行第 3 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 20 ......
词库中包含 128 个单词
precision( 3 ) = 0.9752459784476026 , recall( 3 ) = 0.8761157826024821 , accuracy( 3 ) = 0.966 , f1( 3 ) = 0.9177241532847422
正在执行第 4 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 30 ......
词库中包含 74 个单词
precision( 4 ) = 0.9720075112450326 , recall( 4 ) = 0.8757314628484467 , accuracy( 4 ) = 0.9653333333333334 , f1( 4 ) = 0.9163176325185389
正在执行第 5 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 40 ......
词库中包含 58 个单词
precision( 5 ) = 0.9766321016503579 , recall( 5 ) = 0.9033696538032205 , accuracy( 5 ) = 0.9726666666666667 , f1( 5 ) = 0.9357521852092128
正在执行第 6 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 50 ......
词库中包含 45 个单词
precision( 6 ) = 0.9707407407407407 , recall( 6 ) = 0.8681937744062356 , accuracy( 6 ) = 0.9633333333333334 , f1( 6 ) = 0.910829994412013
正在执行第 7 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 60 ......
词库中包含 37 个单词
precision( 7 ) = 0.9656998054210597 , recall( 7 ) = 0.8577592033959189 , accuracy( 7 ) = 0.96 , f1( 7 ) = 0.9019911051482983
正在执行第 8 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 70 ......
词库中包含 28 个单词
precision( 8 ) = 0.9538708241435827 , recall( 8 ) = 0.8465559928775315 , accuracy( 8 ) = 0.9553333333333334 , f1( 8 ) = 0.8902808482764266
正在执行第 9 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 80 ......
词库中包含 23 个单词
precision( 9 ) = 0.9462829566378665 , recall( 9 ) = 0.84076222774132 , accuracy( 9 ) = 0.9526666666666667 , f1( 9 ) = 0.8837304511586015
正在执行第 10 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 90 ......
词库中包含 20 个单词
precision( 10 ) = 0.9362468883271261 , recall( 10 ) = 0.84927519998146 , accuracy( 10 ) = 0.9526666666666667 , f1( 10 ) = 0.8860220186196144
正在执行第 11 次训练及测试，词库中单词在垃圾邮件中出现的总次数 >= 100 ......
词库中包含 18 个单词
precision( 11 ) = 0.9311072216264341 , recall( 11 ) = 0.8291746974688972 , accuracy( 11 ) = 0.9473333333333334 , f1( 11 ) = 0.8706296569229 51

模型评估完毕
```

图 3-12 SVM 模型的评估结果

```
正在执行测试.....
测试完毕

accuracy =  0.9846666666666667 , precision =  0.9714442462228257 , recall =  0.9613652428166969 , f1 =  0.9663267128722728
```

图 3-13 朴素贝叶斯模型的评估结果

3.4.2. Bootstrap Evaluate

使用 Bootstrap 对数据集进行划分后，对两种模型评估的结果如图 3-14 和图 3-15 所示。由图可知，在使用同一数据集的情况下，朴素贝叶斯模型的 accuracy、recall、F1 值都优于 SVM 模型，而 SVM 模型的 precision 则高于朴素贝叶斯模型，总体来说，朴素贝叶斯模型更优。这与在 Simple Evaluate 阶段得到的结论相同。

```
D:\Anaconda3\python.exe D:/PycharmProjects/DataMining/bayes_classify/bootstrap_evaluate.py
正在加载数据，生成词库.....
词库生成完毕

正在生成词矩阵.....
词矩阵生成完成

正在使用bootstrap方法对数据进行划分.....
数据划分完毕

训练集样本数量： 3509 ， 测试集样本数量： 2065

正在计算概率p(s)，p(wi|s)，p(wi|ns).....
概率计算完毕

正在进行测试.....
测试完毕

precision =  0.9493932994879426 , recall =  0.9667057424332524 , accuracy =  0.9786924939467312 , f1 =  0.9577987344264975

Process finished with exit code 0
```

图 3-14 朴素贝叶斯模型的评估结果

```
D:\Anaconda3\python.exe D:/PycharmProjects/DataMining/svm_classify/bootstrap_evaluate.py
正在加载并处理数据.....
数据处理完成

正在使用bootstrap方法对数据进行划分.....
数据划分完毕

训练集样本数量： 3497 ， 测试集样本数量： 2077

正在进行模型训练.....
模型训练完成

Training accuracy =  0.9674006291106663

正在进行模型评估.....
模型评估完成

precision =  0.97954409739872 , recall =  0.8921723337705579 , accuracy =  0.9706307173808377 , f1 =  0.9298028171511237

Process finished with exit code 0
```

图 3-15 SVM 模型的评估结果

3.4.3. K-Fold Evaluate

表 3-1 为朴素贝叶斯模型与 SVM 模型在 10 轮交叉验证中的评估结果，以及经过计算得到的真实评估结果（平均值）。由表可知，在几乎每一轮的训练中，朴素贝叶斯模型的 accuracy、recall、f1 值都高于 SVM 模型，而 SVM 模型的 precision 都高于朴素贝叶斯模型，总体来说，朴素贝叶斯模型更优。这与在 Simple Evaluate 阶段得到的结论相同。

表 3-1 朴素贝叶斯与 SVM 的 10 轮评估结果

	Module	Precision	Recall	F1	Accuracy
第 1 轮	Bayes	0.946	0.978	0.961	0.980
	SVM	0.987	0.919	0.949	0.977
第 2 轮	Bayes	0.962	0.971	0.967	0.982
	SVM	0.979	0.881	0.921	0.962
第 3 轮	Bayes	0.942	0.970	0.956	0.980
	SVM	0.982	0.870	0.916	0.968
第 4 轮	Bayes	0.941	0.977	0.958	0.980
	SVM	0.992	0.944	0.966	0.986
第 5 轮	Bayes	0.932	0.942	0.937	0.971
	SVM	0.974	0.881	0.921	0.968
第 6 轮	Bayes	0.940	0.969	0.953	0.980
	SVM	0.988	0.908	0.943	0.978
第 7 轮	Bayes	0.936	0.956	0.946	0.975
	SVM	0.976	0.896	0.931	0.971
第 8 轮	Bayes	0.962	0.976	0.969	0.984
	SVM	0.975	0.897	0.931	0.968
第 9 轮	Bayes	0.924	0.967	0.944	0.973
	SVM	0.964	0.875	0.913	0.964
第 10 轮	Bayes	0.958	0.980	0.969	0.986
	SVM	0.982	0.875	0.919	0.968
平均值	Bayes	0.944	0.969	0.956	0.979
	SVM	0.980	0.894	0.931	0.971

4.总结

在本实验中，无论是采用 Bootstrap 方法还是 K-Fold 方法进行评估，都发现朴素贝叶斯模型的 accuracy、recall、f1 值高于 SVM 模型，而 SVM 模型的 precision 高于朴素贝叶斯模型，总体来说，朴素贝叶斯模型表现更好。

本实验中采用词向量来表示邮件的特征向量，这种方法还有可改进之处，例如单词的不同种变型是否可以看作是同一个单词（如 play，playing，plays 等）。

在实验过程中也遇到了一些问题：例如绘制出的 ROC 曲线下方面积的值，即 AUC < 0.5，这是因为 ROC 中要求衡量的是模型将样本预测为正的概率，但实际在计算 TPR 和 FPR 的过程中，可能得到的是模型将样本预测为负的概率，从而导致 AUC < 0.5。

通过本次实验，我对朴素贝叶斯模型和线性 SVM 模型有了进一步的理解，对 Bootstrap、K-Fold、ROC 曲线等评估方法也有了进一步掌握，之后我会尝试更多数据挖掘算法的实现，加深记忆和理解。